**DEPARTMENT OF INFORMATION TECHNOLOGY**

**COURSE CODE: DJS22ITL504**  **DATE: 4/08/24**

**COURSE NAME: Cryptography and Network Security Laboratory** **CLASS: TYBTech**

**NAME: Anish Sharma**

**EXPERIMENT NO. 1**

**CO/LO:** Design secure system using appropriate security mechanism

**AIM / OBJECTIVE:**

1. Implementation of Ceaser Cipher on alphanumeric data.
2. Implementation of Ceaser Cipher on gray scale image.

.

**THEORY / CONCEPT / ALGORITHM:**

The Caesar Cipher is a basic encryption technique in which each letter in the plaintext is shifted a fixed number of positions down or up the alphabet. The shift value, known as the key, is the same for both encryption and decryption. For example, with a shift of 3, 'A' becomes 'D', 'B' becomes 'E', and so on. If the end of the alphabet is reached, the cipher wraps around to the beginning

The Caesar Cipher can be adapted for grayscale image encryption by treating each pixel value as a character in the algorithm. In a grayscale image, each pixel intensity ranges from 0 to 255. Encryption involves shifting each pixel's intensity by a fixed key value, wrapping around using modulo 256 arithmetic. Mathematically, for a pixel value ppp, encryption is E(p)=(p+k)mod  256E(p) = (p + k) \mod 256E(p)=(p+k)mod256 and decryption is D(p)=(p−k)mod  256D(p) = (p - k) \mod 256D(p)=(p−k)mod256, where kkk is the key. This technique provides basic encryption, making the image appear noisy and obscured, though it is not highly secure.

**SOURCE CODE:**

**a)** Implementation of Ceaser Cipher on alphanumeric data.

def encrypt\_decrypt\_string(input\_string, key):

    key = int(key) % 256

    encrypted\_decrypted\_chars = [chr(ord(char) + key) for char in input\_string]

    return ''.join(encrypted\_decrypted\_chars)

def decrypt\_string(input\_string, key):

    key = int(key) % 256

    encrypted\_decrypted\_chars = [chr(ord(char) - key) for char in input\_string]

    return ''.join(encrypted\_decrypted\_chars)

def main():

    input\_string = input("Enter the string to be encrypted/decrypted: ")

    key = input("Enter a numeric key for encryption and decryption: ")

    encrypted\_string = encrypt\_decrypt\_string(input\_string, key)

    print(f"Encrypted string: {encrypted\_string}")

    decrypted\_string = decrypt\_string(encrypted\_string, key)

    print(f"Decrypted string: {decrypted\_string}")

if \_\_name\_\_ == "\_\_main\_\_":

    main()

b) Implementation of Ceaser Cipher on gray scale image

from PIL import Image

import numpy as np

def load\_image(image\_path):

    image = Image.open('grayscale.jpeg').convert('L')

    return np.array(image)

def save\_image(image\_array, output\_path):

    image = Image.fromarray(image\_array.astype(np.uint8))

    image.save(output\_path)

def encryp  t\_decrypt\_image(image\_array, key):

    key = int(key) % 256

    encrypted\_image\_array = np.bitwise\_xor(image\_array, key)

    return encrypted\_image\_array

def compute\_difference\_image(original\_image, decrypted\_image):

    difference\_image = np.abs(original\_image - decrypted\_image)

    return difference\_image

def main():

    input\_image\_path = 'grayscale.jpeg'

    output\_encrypted\_image\_path = 'encrypted.jpeg'

    output\_decrypted\_image\_path = 'decrypted.jpeg'

    difference\_image\_path = 'difference.jpeg'

    key = input("Enter a numeric key for encryption and decryption: ")

    image\_array = load\_image(input\_image\_path)

    encrypted\_image\_array = encrypt\_decrypt\_image(image\_array, key)

    save\_image(encrypted\_image\_array, output\_encrypted\_image\_path)

    print(f"Encrypted image saved as {output\_encrypted\_image\_path}")

    decrypted\_image\_array = encrypt\_decrypt\_image(encrypted\_image\_array, key)

    save\_image(decrypted\_image\_array, output\_decrypted\_image\_path)

    print(f"Decrypted image saved as {output\_decrypted\_image\_path}")

    difference\_image\_array = compute\_difference\_image(image\_array, decrypted\_image\_array)

    save\_image(difference\_image\_array, difference\_image\_path)

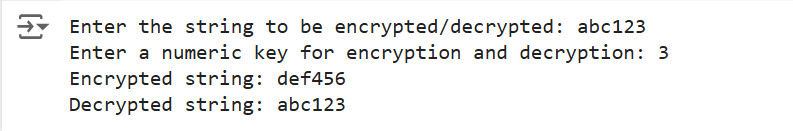
    print(f"Difference image saved as {difference\_image\_path}")

if \_\_name\_\_ == "\_\_main\_\_":

    main()

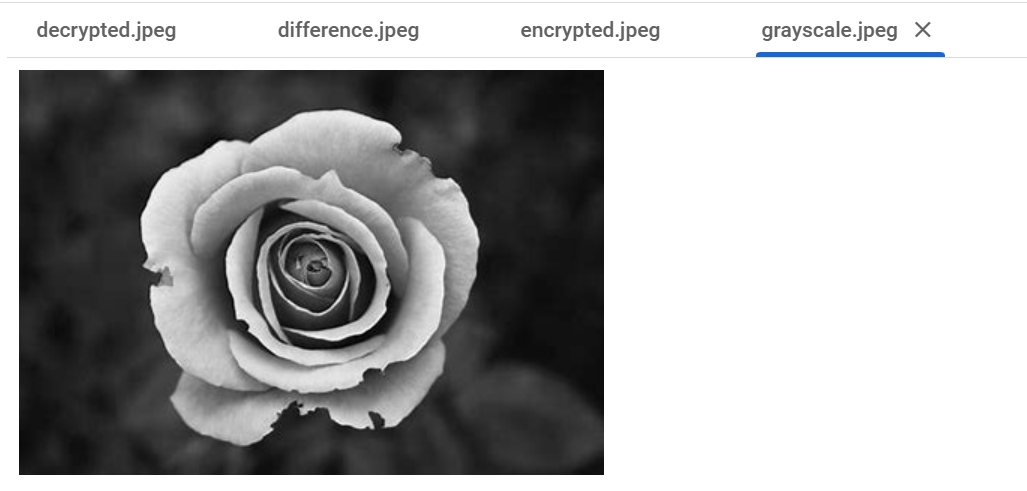
**SAMPLE INPUT AND OUTPUT:**

**a)** Implementation of Ceaser Cipher on alphanumeric data.

****

**b**) Implementation of Ceaser Cipher on gray scale image

**Original Image**

****

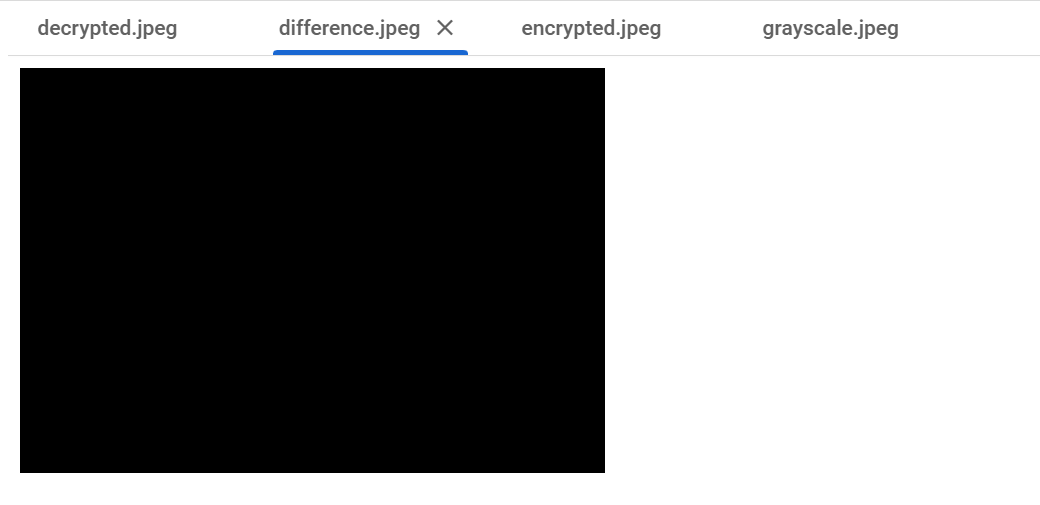
**Encrypted image**

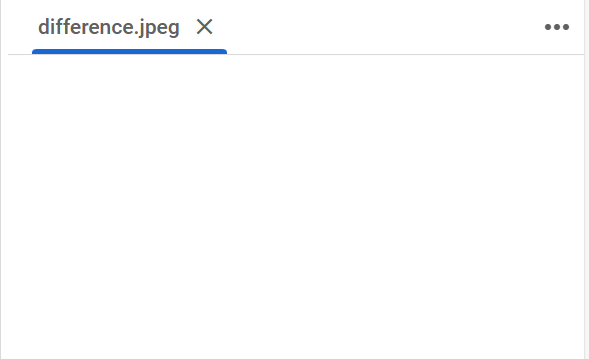
****

**Decrypted image**

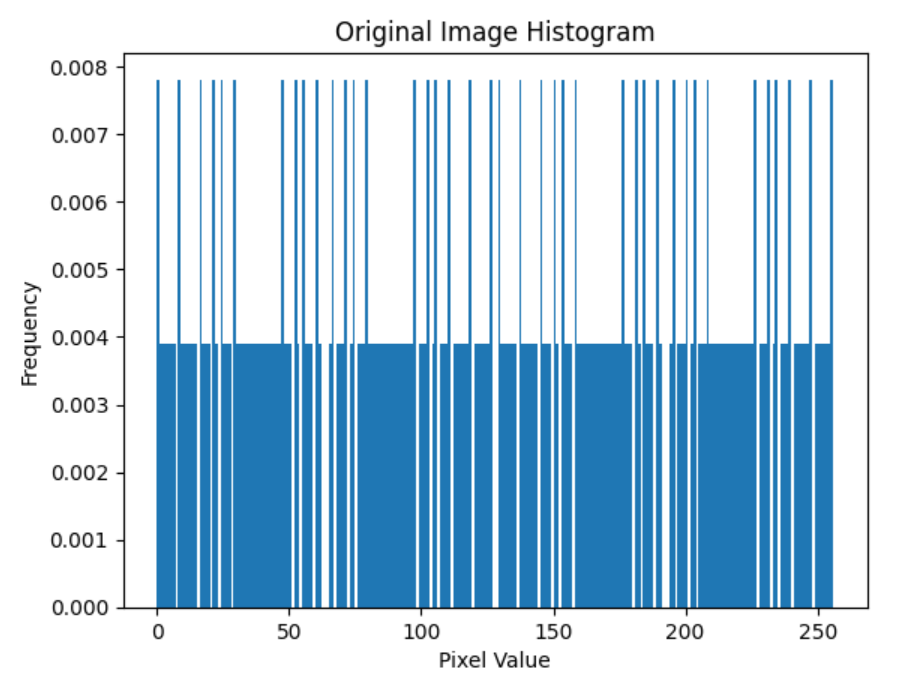
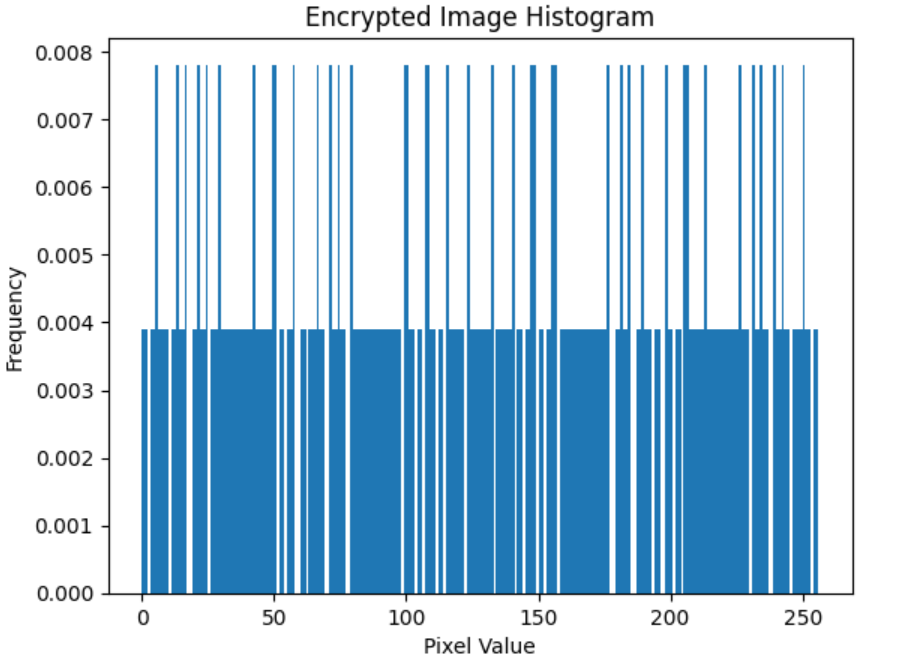
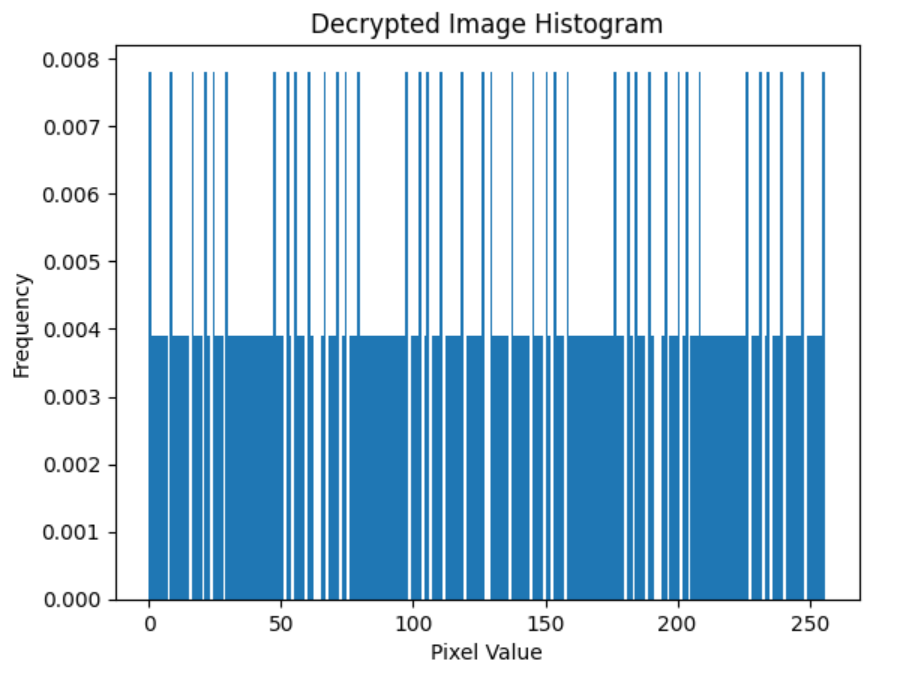
****

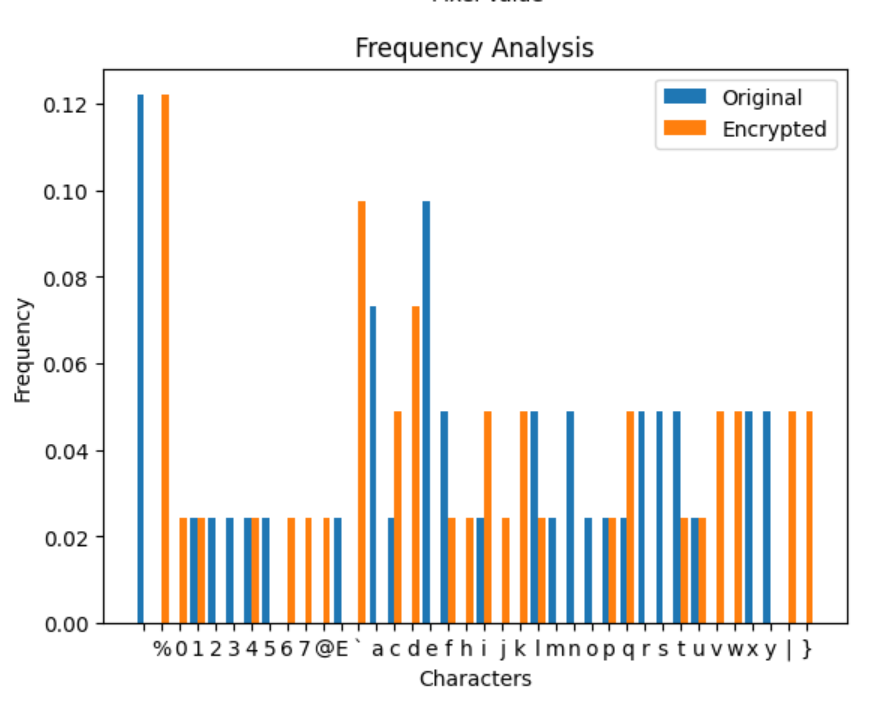
**Difference images**

****

****

**QUESTIONS:**

1. Perform a frequency analysis of the encrypted alphanumeric data and compare it to the frequency of the original data.
2. Generate histograms of the pixel values for the original, encrypted, and decrypted images.**** **** 

****

**CONCLUSION:** We performed Ceaser Cipher encryption and decryption algorithms on text and grayscale images

**REFERENCES:**

**Pillow (PIL)**:

* Official documentation: [Pillow Documentation](https://pillow.readthedocs.io/)